**VŠB – Technická univerzita Ostrava Fakulta elektrotechniky a informatiky Katedra informatiky**

# Automatický asistent korektora textů metodami strojového učení

An Automatic Text Corrector's Assistant Using Machine Learning Methods

2018 Tomáš Walek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání bakalářské práce

Student: **Tomáš Walek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Automatický asistent korektora textů metodami strojového učení

An Automatic Text Corrector's Assitant Using Machine Learning Methods

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit program sloužící jako pomocný nástroj při korekturách textů. Řešení bude založeno na metodách analýzy přirozeného jazyka. Student využije korpus existujících textů, u kterých již proběhla korektura. V řešení lze využít i prvky strojového učení.

Bakalářská práce musí splňovat následující body:
1. Stručný přehled metod pro klasifikaci dat.
2. Přehled podobných existujících řešení.
3. Výběr metody vhodné k řešení problému a zdůvodnění volby konkrétní metody.
4. Popis a implementace zvolené metody.
5. Vyhodnocení výkonnosti řešení.

Seznam doporučené odborné literatury:

[1] Salton, Gerard. Automatic text processing: The transformation, analysis, and retrieval of. Reading: Addison-Wesley (1989).
[2] Stuart J. Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Pearson Education, 2003.
Dále podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Dohnálek**

Datum zadání: 01.09.2016

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
*vedoucí katedry*

prof. Ing. Pavel Brandštetter, CSc.
*děkan fakulty*

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. června 2018 .....

## Poděkování

Rád bych poděkoval **Ing. Pavlu Dohnálkovi** za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

## Abstrakt

Cílem bakalářské práce bylo vytvořit algoritmus, který bude sloužit jako nástroj pro automatickou korekturu textu. Algoritmus použitý v této bakalářské práci vychází z metod analýzy přirozeného jazyka. Řešení pro automatickou korekturu textu bylo inspirováno prvky strojového učení. V této práci je využit korpus existujících textů, u kterých již proběhla korektura.

**Klíčové slova:** Automatická korektura textu, strojové učení, přirozený jazyk

## Abstract

The purpose of this bachelor thesis was to create and algorithm, which will serve as tool for automatic text correction. Algorithm used in this bachelor thesis derives from methods of natural languages analysis. Solution for automatic text correction was inspired by elements from machine learning. There is a corpus of already proof read texts used in this thesis.

**Key words:** Automatic text correction, machine learning, natural language

# 1 Obsah

# Seznam obrázků

# Úvod

Oblast IT zaznamenala v posledních letech velký posun vpřed. Samotný textový obsah se rozšiřuje zejména v elektronické podobě. Takové texty můžeme pak nalézt především v podobě novinového článku, odborné publikace, v uživatelském obsahu či ve formě recenze. Samotný text je povětšinou napsán v přirozeném jazyce, v důsledku čehož často roste i procento chybně napsaných slov či vět, které jsou v textu obsaženy.

Tyto chyby mohou vznikat z různých důvodů. Hlavní příčinou jsou špatný pravopis, překlepy, nesprávná gramatika, či jiné jazykové jevy. V situacích, které vyžadují opravu takovéto chyby, se nejčastěji můžeme setkat s aktivní korekturou textu, jež nás upozorňuje na jednotlivé chyby či překlepy, ale která zároveň vyžaduje interakci autora či korektora. Zároveň nám také nabídne možnost nahrazení slova, na které jsme byli předtím upozorněni, podobným výrazem, který je obsažený ve slovníku. Tyto aktivní metody však mohou být nevyhovující, jelikož jednotlivá slova vyžadují interakci zainteresované osoby, která musí zpracovat a opravit velké množství dat v podobě textu.

Vedle aktivní korektury textu s interakcí autora či korektora se také vyvíjejí metody automatické korektury textu, které ale nevyžadují onu aktivní osobní účast. Program k automatické korektuře textu tedy může ve výsledku zpracovat text a opravit jej za mnohem kratší čas než fyzická osoba. Další nespornou výhodou vedle úspory času tedy může být možnost člověka nezaměstnávat, a tedy neplatit osobě, jejíž náplní práce by byla právě korektura textu.

Tato bakalářská práce se zabývá automatickou korekturou textu, kterou se snaží vyřešit ony nevýhody aktivní korektury. Cílem mé bakalářské práce je navrhnout a implementovat algoritmus, který by korekturu textu zautomatizoval a poskytl autorům textu asistenční nástroj pro samostatnou korekturu textu.

V úvodu bakalářské práce byla tedy shrnuta problematika korektury textu. Dále jsme uvedli, že existují aktivní metody korektury textu, které vyžadují interakci fyzické osoby, a vedle nich metody nevyžadující osobní interakci. Měli bychom si také objasnit, co je to jazyk, k čemu slouží a jaké druhy jazyků známe. Ve druhé kapitole této bakalářské práce si nastíníme, co vůbec znamená pojem „korektura textu“ a jaké jazykové jevy mohou být opraveny. Zároveň v této kapitole rozvineme již existující řešení automatické korekce textu, o níž se tato práce zmiňuje v úvodu. Třetí kapitola se věnuje problematice rozhodovacích stromů, jejich popisu a základnímu rozdělení. Čtvrtá kapitola se bude věnovat implementaci vlastního algoritmu a samotné technologii, která byla při vypracování této bakalářské práci použita. V páté kapitole již budeme vyhodnocovat efektivitu a rychlost algoritmu, jež byl pro tuto bakalářskou práci použit.

# 1 Jazyk

Jazyk je velmi úzce spjatý s písmem, tedy s jeho grafickým znázorněním. Již odnepaměti slouží k dorozumívání mezi lidmi nejrůznějšího původu. Kromě funkce dorozumívací má také funkce expresivní či emotivní. Není možné naprosto přesně určit, kolik jazyků na světě existuje, a to hned z několika důvodů; zaprvé může být obtížné naprosto dokonale odlišit jazyk od jazykového dialektu a zadruhé s globalizací vznikají jazyky nové a některé zanikají. Mezi tzv. mrtvé jazyky, tedy jazyky, které již v dnešní době nikdo neužívá v aktivní řeči, patří například latina či sanskrt.

## 1.1 Druhy jazyků

Vedle přirozených jazyků, které vznikly před sty tisíci lety, existují také jazyky umělé, ať už to jsou jazyky fiktivní, jako je např. klingonština, tak i jazyky, které mají základy v jiných, již existujících jazycích. Mezi nejznámější umělé jazyky patří esperanto. To jsou ale stálé jazyky, které se užívají k dorozumívání mezi určitou sortou lidí. Vedle nich existují strojové jazyky, které slouží jako komunikační nástroj mezi programátorem a počítačem; řeč je tedy o jazycích programovacích, které slouží jako prostředek pro zápis algoritmů, které budou na počítači provedeny.

## 1.2 Český jazyk

V této bakalářské práci se budu zabývat algoritmem, který slouží ke korektuře českého jazyka. Čeština jako taková patří do rodiny indoevropských jazyků, konkrétně do její slovanské větve, stejně jako polština, ruština či slovenština. Čeština je jazyk flektivní, to znamená, že má komplikovaný systém skloňování a časování; má také velmi volný slovosled. Český jazyk se vyvíjí již od konce 10. století a jeho vývoj neustále pokračuje. Vliv na tento vývoj měly nejvíce sousedské vztahy s Německem, do své slovní zásoby také čeština přejala slova z řečtiny i latiny, jako je zvykem u většiny evropských jazyků, a později také globalizace, kdy čeština přejímá slova z francouzštiny, italštiny a nejvíce samozřejmě z angličtiny. Existuje velká řada způsobů, jak lze rozšířit slovní zásobu českého jazyka, ať už je to přejímání z jiných jazyků, tvoření slov nových (zejména v době Národního obrození vznikala nová slova, která se však ujala nepříliš úspěšně) či překládáním z cizích jazyků.

# 2 Korektura textu

Samotné slovo „korektura“ lze definovat jako opravu zjištěných chyb, nepřesností či nesrovnalostí v textu. Slovo pochází z latinského *corriegere*, což doslova znamená „opravovat“. Korekturu textu může provádět fyzická osoba, která se nazývá korektor. Pro osobu korektora je nezbytná perfektní znalost jazyka. Chyby obsažené v textu mohou mít různý původ; chyby mohou být pravopisné, morfologické, syntaktické, nebo mohou mít původ i v jiném gramatickém jevu.

## 2.1 Pravopisné chyby

Pravopis můžeme popsat jako ustálený způsob záznamu slov v grafické podobě. Pravopis bývá často kodifikován, v České republice je orgánem činným pro kodifikaci pravopisu Ústav pro jazyk český Akademie věd. Pravopis na území nynější České republiky se dlouhá staletí vyvíjel a měnil. I v dnešní době, kdy ve světě vládne trend globalizace, se můžeme v češtině setkat s novými slovy či pojmy, které přejímáme z cizích jazyků. Vzhledem k tomu, že čeština patří mezi nejsložitější jazyky světa, není samozřejmě ani český pravopis jednoduchý, a osoba, která nemá jazykové cítění, může často ve psaném projevu chybovat. Jako nejčastější pravopisné chyby můžeme určitě zmínit omyly ve psaní „tvrdého“ a „měkkého“ y/i, dále chyby v písmenech s/z na začátku slov či chyby v interpunkci nebo v diakritice. Ve psaní tvrdého „y“ hrají hlavní roli vyjmenovaná slova. To jsou slova, ve kterých se po obojetných písmenech (b, f, l, m, p, s, v, z) píše vždy v jejich kořenu „y“. Také ve slovech jim odvozených píšeme vždy „y“. Samozřejmě zde můžeme nalézt výjimky či slova, která jsou si významově velmi podobná, ale nejsou to slova příbuzná, a tak se v každém z nich píše jiné i/y. Jako příklad uvedu vyjmenované slovo „mlýn“, jehož příbuzné slovo může být např. mlynář nebo mlýnice, a jako protiklad slovo „mlít“, kde by si málokdo pomyslel, že obě slova pocházejí z jiného jazyka, a tak se liší v měkkém a tvrdém i/y. Jako další příklad bych uvedl slova pikat a pykat, kdy obě slova mají rozličný význam, a proto se v nich vždy píše jiné i/y. Kapitolou samou pro sebe je interpunkce. Interpunkční znaménka jsou jakési grafické znaky, které vytvářejí, strukturují a organizují psaný text. Mezi nejzákladnější interpunkční znaménka v českém jazyce patří tečka, která ukončuje větu či označuje zkratky nebo řadové číslovky, dále čárka, jež má za úkol oddělovat věty v souvětích, dvojtečka, která často uvozuje přímou řeč, a uvozovky, které právě přímou řeč označují. I pro tato interpunkční znaménka existuje soubor jakýchsi pravidel, která přikazují, jakým způsobem se připojují k předcházejícímu slovu či jestli je před nimi nebo za nimi mezera a podobně. Příkladem, jak jsou interpunkční znaménka důležitá, může být věta: „Jezte, děti!“, kdy při vynechání interpunkce vznikne zvolání: „Jezte děti!“. Příslušníci divokých kanibalských kmenů by toto jistě ocenili; v českém právním řádu je však kanibalismus přísně trestán, proto bychom zde na čárku neměli zapomínat. [1]

## 2.2 Morfologické chyby

Morfologie neboli tvarosloví se zabývá tvořením slov, jejich skloňováním a časováním, což můžeme souhrnně označit jako ohýbání. Čeština má sedm pádů a každé podstatné jméno se skloňuje podle určitého vzoru. Zejména při skloňování slov, které jsou v množném čísle, mohou vznikat chyby v koncovkách slova. Uvedu příklad: sedmý pád množného čísla slovního spojení „dvě ruce“ by měl být správně napsaný „se dvěma rukama“; nezřídka se však setkáme s podobou „se dvouma rukami“. Co se týče časování, zde bývá opět nejčastější chyba ve psaní koncovky, tentokrát v příčestí minulém. Český jazyk u podstatných jmen rozlišuje rody ženské, mužské a střední, což má za následek rozdílné koncovky u sloves, která se k daným podstatným jménům vztahují. Zatímco u podstatného jména v rodě mužském a čísle množném bude u slovesa v minulém čase koncovka „i“, v rodě ženském to pak bude „y“ a v rodě středním dokonce „a“.

## 2.3 Syntaktické chyby

Syntax můžeme označit jako jazykovědný obor, který se zabývá větnou skladbou a vztahem mezi slovy ve větě. Syntax se tedy zabývá slovosledem a větnými členy. Český jazyk nemá pevný slovosled, jako tomu bývá například u germánských jazyků, jako je angličtina či němčina. Znamená to tedy, že větné členy v českém jazyce nemají své pevné místo ve větě, ale mohou se vyskytovat na různých pozicích. Přesto však nemůžeme každý větný člen dát na jakékoliv místo ve větě bez úvahy. Opět uvedu příklad; tentokrát mi za vzor poslouží slavná věta Mistra Yody z kultovní série Star Wars: „Vždy v pohybu budoucnost je.“ Syntakticky správně by tato věta samozřejmě zněla: „Budoucnost je vždy v pohybu.“

## 2.4 Kontext

Korektor při opravě souvislého textu pracuje s informační hodnotou (kontext), kterou využívá pro rozhodování v případě více možností. Jako příklad si můžeme uvést větu: „Pepa odešel síl.“ Pokud se trošku zamyslíme, zjistíme, že pro danou větu jsou možné dva způsoby korekce. „Pepa odešel dál,“ nebo „Pepa odešel sám“; obě varianty jsou pravopisně správně. Pokud ovšem budeme trošku kreativní, můžeme uvážit, že slovo „odešel“ je autorův překlep pro slovo „obešel“, a místo překlepu „síl“ mělo být slovo „sál“. Což nám aktuálně dává celkem tři možnosti, které korektor musí zvážit. Pokud ovšem k výše uvedené větě nemáme zbylý kontext, pak ani sám korektor není schopen rozluštit, co autor zamýšlel napsat.

Jak poznat, že je věta špatně a potřebuje korekci? Pokud budeme pracovat s větou „Pepa obešel sál.“, tak ačkoli je věta po gramatické a pravopisné stránce správně, autor mohl v této větě udělat chybu a původně zamýšlet větu „Pepa odešel sám.“ Bohužel tady jsme bez kontextu taktéž bezradní.

## 2.5 Algoritmy zabývající se korekcí

„V suoivsoltsi s vzýukemm na Cmabridge Uinervtisy vlšyo njaveo, že nzeáelží na pořdaí psiemn ve solvě. Jedniná dleůitžá věc je, aby blyy pnvrí a psoelndí pímesna na srpváénm mstíě. Zybetk mžůe být totánlí směs a ty to přoád bez porlbméů peřčetš. Je to potro, že ldiksý mezok netče kdažé pensímo, ale svolo jkao cleek. Zjíamvaé, že?“ [2]

Toto je překlad článku, který tvrdí, že výzkumný tým Univerzity v Cambridgi pracoval na přezkumu toho, jak náš mozek vnímá překlepy. Otázkou však zůstává, jak se s takovými překlepy dokáže vyrovnat technické zařízení. Automatická oprava těchto překlepů často bývá součástí nástrojů jak vyhledávačů (např. Google), či programů používaných při tvorbě dokumentů (typicky Microsoft Word). Co se týče mozku, ten se dokáže s překlepy vypořádat jednoduše; toto však ani zdaleka neplatí u výše zmíněných nástrojů používaných k automatické opravě textu. Překlepy se softwarově řeší různými algoritmy, které ale pracují vždy na podobném principu. V této bakalářské práci si představíme dva z nich. [3]

### 2.5.1 Fonetické shody

Na korekturu textu byla aplikována spousta přístupů. Jeden z těchto přístupů se snaží porovnávat řetězce na základě jejich fonetické shody. Většina těchto algoritmů byla vytvořena pouze pro jazyk anglický. Jako příklad zde uvedu dva z nich, a to Soundex a Metaphone.

Algoritmus Soundexu byl používaný hlavně pro sestavování rodokmenů vzniklé zkomolením příjmení. Tento algoritmus přiřazuje číselné kódy jednotlivým znakům a na základě fonetiky je pak přiřazuje do skupin, které reprezentují fonetické znění daného znaku. Výsledkem jsou čtyřmístné kódy skládající se ze začátečního písmena a tří čísel. Díky přiřazováním podobně znějících slov stejné hodnoty se Soundex nesetkal s praktickým využitím, ačkoli přináší určité výhody.

| **Slovo** | **Kód** |
|---|---|
| Allan | A450 |
| Smith | S530 |
| Davis | D120 |

*Tabulka 1: 3 příklady příjmení a jejich kódů, které jim Soundex přiřadil.*

Druhý základní algoritmus zakládající se na fonetické shodě je Metaphone. Tento algoritmus vznikl, aby odstranil problémy, které vytvářel algoritmus Soundex. Na základě této skutečnosti vzniklo mnoho pravidel, která měla zajistit, aby nevznikaly stejné kódy pro podobně foneticky znějící výrazy. Jako jedno z pravidel bylo například vynechání písmena b, za předpokladu, že na předešlé pozici se nachází písmeno *m*. [4]

| **Slovo** | **Kód** |
|---|---|
| James Wylie | JMSL |
| Jane Milne | JNMLN |
| Janet Brown | JNTPRN |

*Tabulka 2: Ukázka generovaných kódů v algoritmu Metaphone.*

### 2.5.2 N-gramy

N-gramy můžeme nejjednodušeji chápat jako sled **n** po sobě jdoucích znaků. N-gramy se skládají ze slov nebo písmen. Na základě velikosti N-gramu je dělíme na uni-gramy, bi-gramy, tri-gramy atd. Například řetězec *konec* se skládá z těchto N-gramů:

- Bi-gram: -k, ko, on, ne, ec, c-
- Tri-gram: -ko kon one nec ec-
- Quad-gram -kon kone onec nec-

Jestliže bychom chtěli tento přístup aplikovat jako korekturu slov, potřebovali bychom nejdříve získat záznamy n-gramů, které jsou často měněny. Jako následující krok bychom například použili na nekorektní slovo všechny operace pro editaci textu a následně ho rozdělili na n-gramy, které bychom srovnali s n-gramy kandidátů. Rozdílnost dvou slov by se tvořila na základě součtu stejných n-gramů, ze kterých se tyto slova skládají. [5]

### 2.5.3 Podobnosti řetězců

Pro základ tohoto přístupu je třeba zavést určitou metriku, která definuje tuto podobnost. Tato metrika se nazývá Damerau-Levenshteinova vzdálenost.

$$d_{ij} = min \begin{cases} d_{i-1,j} + c_{del}(b_i) \\ d_{i,j-1} + c_{ins}(a_j) \\ d_{i-1,j-1} + [a_j \neq b_i] \times c_{sub}(a_j, b_i) \end{cases}$$

*Rovnice 1: Vyjádření Levenstheinovy vzdálenosti.*

Levensteinova vzdálenost neboli *minimální editační vzdálenost* určuje nejmenší počet editačních kroků, které je nezbytné vykonat k přeměně jednoho řetězce na druhý. Editační krok chápejme jako nahrazení, odstranění nebo záměnu sousedících znaků. Výsledkem Levensteinova algoritmu je celé číslo, které zobrazuje, kolik těchto kroků je třeba k přeměně jednoho z řetězců, aby byl ekvivalentní druhému řetězci.

| **První řetězec** | **Druhý řetězec** | **Výsledek Lev. algoritmu** |
|---|---|---|
| Korekce | Korakce | 1 |
| Praha | Brno | 4 |
| Kniha | Kniha | 0 |

*Tabulka 3: Tabulka zobrazující metriku Levenstheinovy vzdálenosti mezi dvěma řetězci.*

Zjištění velikost rozdílů mezi vstupními řetězci nemusí být nutně použito pouze pro porovnání dvou slov, ale můžeme díky tomuto porovnávat i dvě věty a zjišťovat tak, jak rozdílné jsou. Což se v případě automatické korekce může hodit. Jak již bylo naznačeno, automatická korekce pracuje téměř vždy s vícerem možných výsledků a musí z nich vybrat tu nejvhodnější, a tak se nám přímo vybízí použít tuto metriku k zjištění rozdílů mezi originální větou a možnými opravami. [6]

### 2.5.4 Kontextové okolí

V této práci byly doposud zmíněny metody pracující pouze s izolovanými slovy. Nebereme-li v úvahu, do jaké míry je kvalitní implementace nebo efektivita zmíněných metod, pokaždé zůstává skupina chyb, které tyto metody neodhalí, a tedy ani neopraví. Do těchto skupin patří zejména překlepy; jako příklad si můžeme uvést slovní spojení *dostihový tůně,* kdy překlepem ve slově *kůň* vzniklo taktéž pravopisně správné slovo, které ale nedává smysl v daném kontextu. Metody kontextového okolí mají tedy za úkol vybrat korektní slovo z množiny slov, u kterých je možná záměna z důvodu jejich podobnosti. V případě *dostihový tůň* vytvoříme množinu {*vůně, tůně, lůně*}. Způsobů, kterými lze řešit tyto problémy, je několik; například rozhodovací stromy, Bayesova metoda či využití trigramů.

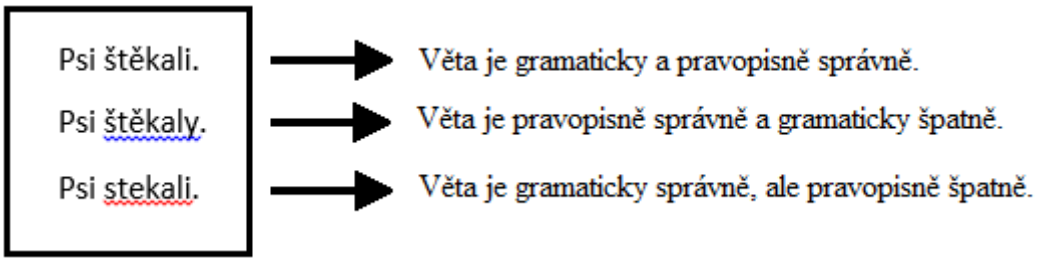
## 2.6 Programy s korekcí

Jak bylo zmíněno, existují dva typy automatické korekce. Celková, to je ta, která má na vstupu text a jejíž výstup je text po korekci, a poloviční – ta má na vstupu text a její výstup je stejný text s upozorněními u každého slova nebo fráze, které jsou vyhodnoceny algoritmem jako nesprávné, a autorovi jsou v určitých případech i nabídnuty možnosti, jak danou nesprávnost opravit.

Tato poloviční automatická korekce má obrovský vliv na výsledek, protože sám korektor jakožto osoba může vyhodnotit, které z různých možností do kontextu zapadají, a jestli je vůbec potřeba zasahovat do textu, ačkoliv k tomu poloautomatická korekce vybízí. Za příklad si můžeme vzít software Microsoft Word, který slova, která mu nevyhovují, podtrhuje a nabízí autorovi nebo korektorovi možnosti za lepší volbu, ale rozhodnutí, jestli volba využita bude nebo nebude, záleží vždy na autorovi. Výsledný text je tedy vždy stejně kvalitní jako autorova (korektorova) znalost daného jazyka.

### 2.6.1 Microsoft Word

Korekce textu v programu Microsoft Word je poloautomatická. Tedy aplikace kontroluje průběžně autorův text, který autor píše, a provádí dozor nad tím, zda text neobsahuje chyby. Kontrola chyb v aplikaci Microsoft Word je rozdělena na dvě části. První část se zabývá kontrolou správnosti jednotlivých slov izolovaně od zbytku kontextu a druhá část se zabývá gramatikou. V aplikaci Microsoft Word jsou chyby rozděleny a označeny červeným a modrým podtrhnutím, jak můžeme vidět na Obrázek 1.




*Obrázek 1: V aplikaci Microsoft Word jsou chyby rozdělení na červené a modré podtrhnutí.*

Gramatické chyby jsou odhaleny na základě lingvistických pravidel. Tato pravidla vylučují konstrukce, které v dané větě být nemůžou, např.: předložka nebo spojka a sloveso nemůžou následovat ihned po sobě.

Tento přístup je schopný odhalit až 40 % chyb, kterých se autor dopustí. Občas však nastane situace, kdy tento systém upozorní na chybu, která ve skutečnosti chybou není. [7]

### 2.6.2 Aspell

Způsoby zabývající se korekcí textu zakládající se na principu fonetické shody nebo srovnávání řetězců jsou nedostačující. Algoritmus, jehož celý název zní GNU Aspell, tyto dva principy kombinuje. Tento software kontroluje, zdali jsou slova ve slovníku. Pokud slova nejsou ve slovníku, software je považuje za překlep a vygeneruje seznam návrhů na opravu. Algoritmus vygeneruje slova se stejným fonetickým kódem a editační vzdáleností, která nepřekračuje definovanou mez. Dalším krokem je vyhledání fonetického kódu slov, která nepřekračují editační vzdálenost 1 v porovnání s chybným slovem. Podle těchto kódů jsou přidána slova s totožným kódem jako slovo chybné. Pokud nebyla nalezena žádná náhrada za chybné slovo, je vytvořen seznam všech slov ze slovníku mající stejný fonetický kód s nejmenší editační vzdáleností od chybného slova. Finální krok seřazuje vybraná slova dle editační vzdálenosti. Algoritmus v softwaru Aspell nepoužívá pro editační vzdálenost Levenstheinovu vzdálenost, která je zaměřena pouze na nahrazení, vložení nebo odstranění písmen ve slově, ale jsou navíc zahrnuty operace záměny sousedících znaků a změna ve velikosti písmen. [4]

### 2.6.3 Ispell

Tento korektor byl původně vytvořený pro Unixový systém s podporou velké části západních jazyků. Ispell nabízí několik rozhraní, jako například programové (použité v textovém editoru GNU Emcs). Tento směr se později ujal i v dalších aplikacích, které později začaly implementovat nástroje na korekci textu do svých rozhraní. Jako většina algoritmů pro korekci textu i Ispell prochází jednotlivá slova v textu a porovnává je se slovy, která jsou uložena ve slovníku. Pokud nenalezne shodu, vygeneruje možné opravy pro takové slovo a nabídne je autorovi textu i s chybným slovem.

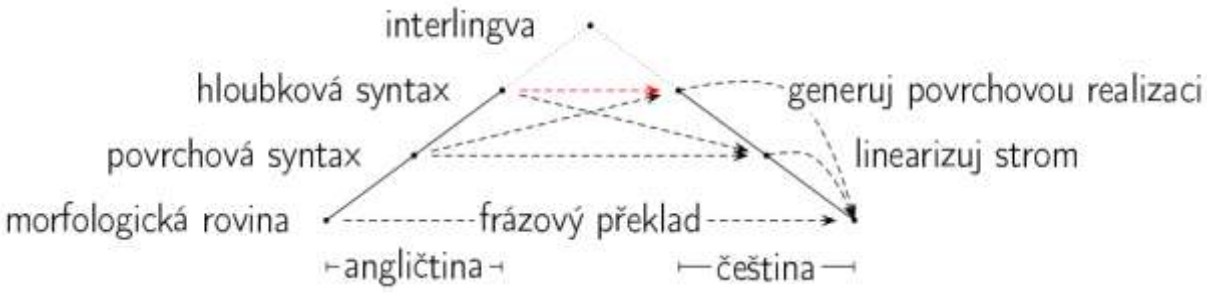
Hlavní rozdíl mezi aplikacemi Ispell a Aspell je v rozdílu použité editační vzdálenosti při generování náhrad, kdy Ispell nepřesahuje editační vzdálenost 1. [4]

### 2.6.4 Hunspell

Korektor Hunspell byl původně navržen pouze pro jazyk maďarský. Vychází z aplikace na korekturu textů Myspell, která byla integrována v balíku OpenOffice, než ji Hunspell nahradil. V Hunspellu je možné dále korektorovat spoustu jazyků, které jsou velice bohaté tvaroslovím a skladbou slov. Hunspell používá rozdělené slovníky, kdy v jednom souboru jsou uloženy všechny slova (soubor s příponou *.dic*) a v druhém souboru (přípona *.aff*) jsou pak uloženy všechny předpony a přípony, na které se slovník odkazuje. Aktuálně je využíván v programech, jako je například Google Chrome, OpenOffice, Opera nebo Firefox. [4]

## 2.7 Překladače

Pokud bychom automatickou korekci chápali jako druh jednojazyčného překladu, pak bychom díky této analogie mohli chápat zdrojovou větu jako chybnou a cílovou větu jako větu, která by měla mít správnou gramatiku a pravopis bez změny v informativní hodnotě, kterou nese zdrojová věta.



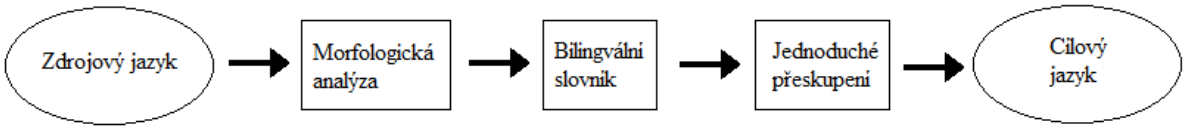

*Obrázek 2:Obrázek popisující přístup ke strojovému učení. Čím více je vstup rozebrán, tím snazší by měl být překlad.*

### 2.7.1 Lingvisté proti statistikům

Chápání strojového překladu a způsob, jakým k němu přistupovat, se rozdělil na dva tábory. První tábor pohlížel na strojový překlad čistě z matematického hlediska. Dle Warren Weavera lze přistupovat k překladu čistě jako k dešifrovací úloze. „Mám ruský text, budu však předstírat, že je ve skutečnosti napsán anglicky a jen zašifrován do neznámých symbolů. Stačí tu šifru rozluštit.“ Druhý tábor, jehož stoupencem byl např. Noam Chromsky, přistupoval ke strojovému překladu čistě z lingvistického pohledu. „Pojem, pravděpodobnost věty‘ je zcela k ničemu, a to při jakékoli známé interpretaci.“

### 2.7.2 Přímý překlad

První strojové překlady vznikaly na přelomu 50. a 60. let minulého století, kdy se neuplatňovaly žádné mezičlánky. Model přímého překladu je znázorněn na Obrázek 3.




*Obrázek 3: Přímý překlad Hutchins a Somers 1992*

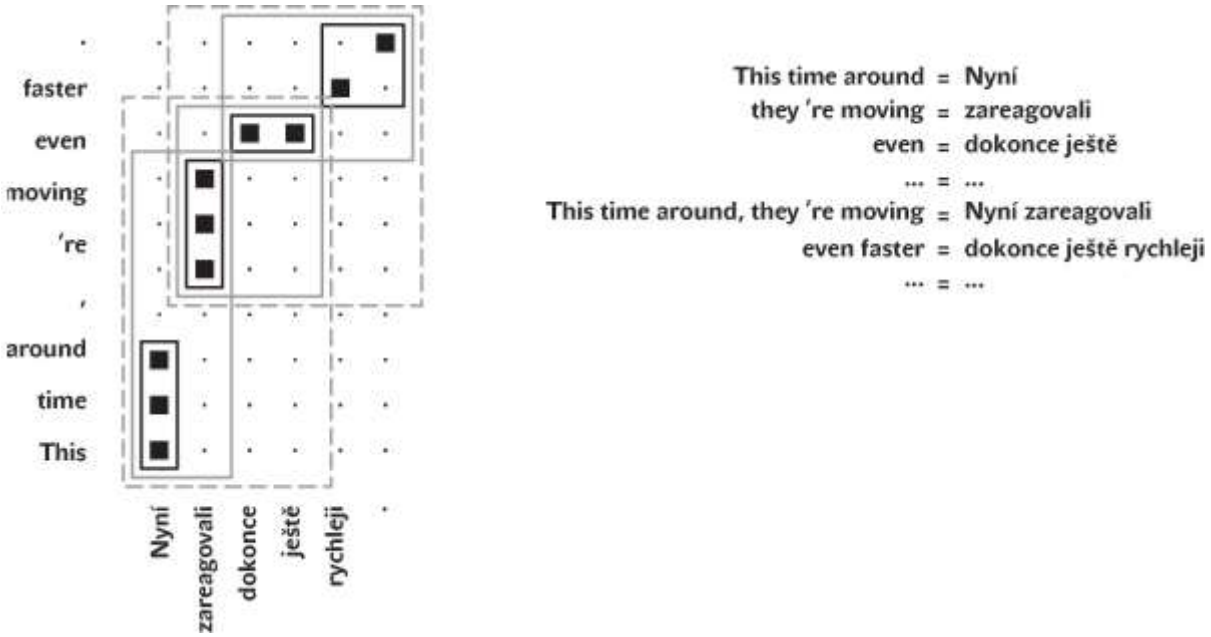
Na Obrázek 3 můžeme vidět znázornění obecného modelu přímého překladu. Byla provedena analýza zdrojového textu z morfologického hlediska, poté systém výsledky analýzy přeložil pomocí slovníku. Konec analýzy spočívá v menším přehození slov na základě toho, jakým způsobem je v cílovém jazyce dána větná skladba. [8]

### 2.7.3 Frázový statistický překlad

V této kapitole se budeme zabývat metodami strojového překladu, které jsou založeny na statistice. Tyto metody si můžeme souhrnně popsat jako situaci, kdy máme k dispozici text v určitém

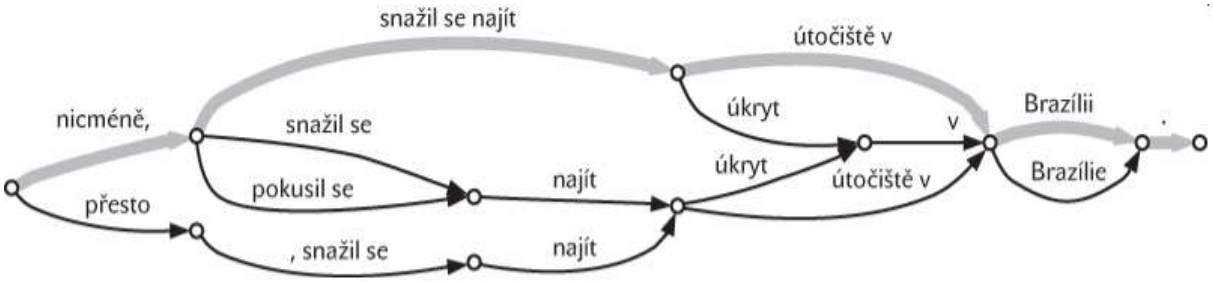
jazyce, který neovládáme, a zároveň naprosto shodný text v nám známém jazyce. Frázový překlad přistupuje ke slovům jako k nedělitelným a izolovaným jednotkám. Algoritmus frázového překladu tedy není schopen vidět spojitost mezi slovy psi a psí a už vůbec ne mezi slovy psi a fena. Věta je brána jako množina znaků, která má být převedena na jinou množinu znaků.

Metoda frázového překladu je založena na velkém množství vět, které byly přeloženy lidmi. Tyto věty nazýváme korpusy. Princip frázového překladu spočívá v práci překladového programu s bilingválním (tedy dvojjazyčným) korpusem, kdy se jednotlivá slova spojovaly se svými ekvivalenty v druhém jazyce, a program v rámci každého páru vyhledá, která slova se shodují. Na místo běžných slovníků tento slovník obsahuje i posloupnosti několika slov. Důležité je, že jsou daná slova uložena ve všech formách, jak byla spatřena. Na Obrázek 4 můžeme vidět způsob vytváření slovníku. [9] [10]




*Obrázek 4: Frázový překlad, vytváření slovníku.*

V prvním kroku je vstupní věta rozložena na několikaslovné úseky ve všech variantách. Program se zaměří na jednu překladovou jednotku v jazyce A a v korpusu vyhledá všechny její možné překlady. Z nich poté program vypočítá, s jakou pravděpodobností se vyskytuje přeložená jako varianta 1, varianta 2 či varianta 3 atd. Vedle toho, s jakou frekvencí se jednotlivé varianty v korpusu vyskytují, zároveň dbá na to, aby byly vybrány takové, které na sebe nejlépe navazují. Následně program přeloží nový text, který upraví do běžné podoby v jazyce B (tedy v cílovém jazyce) pomocí jiných statistických výpočtů. Na Obrázek 5 můžeme vidět výběr nejlepší možnosti frázového překladu.

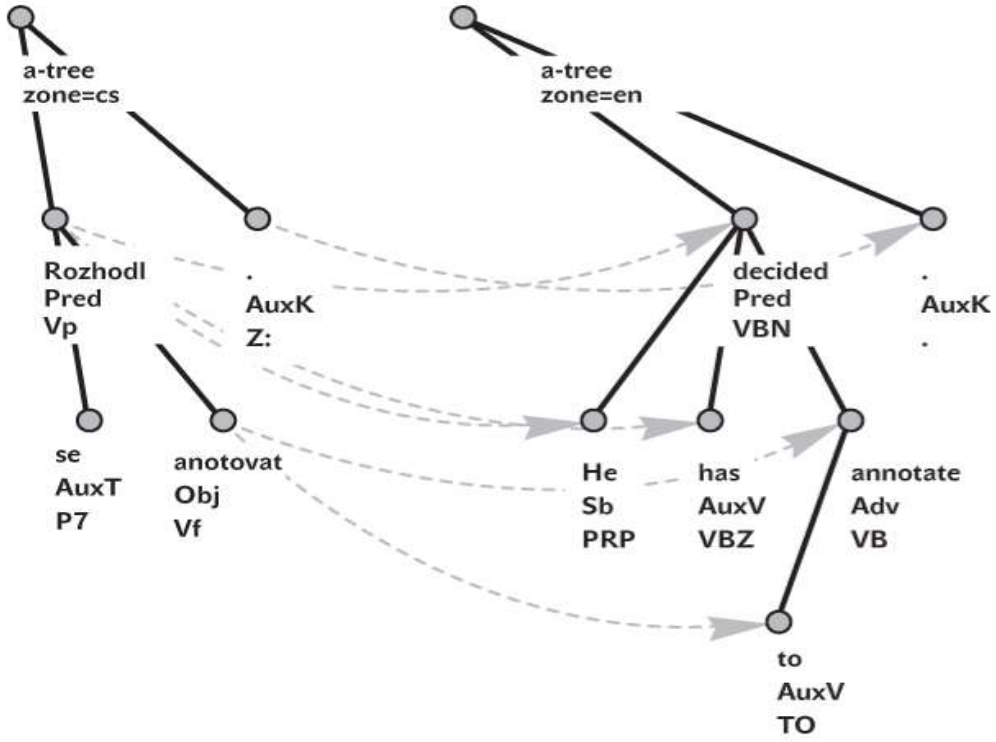



*Obrázek 5: Jednotlivé hrany jsou ohodnoceny pravděpodobností a algoritmus vybírá nejlepší (nejlevnější) cestu.*

Další metodou statistického překladu je tzv. hybridní překlad. Vzhledem k tomu, že žádné dva jazyky na světě nemají naprosto shodná gramatická pravidla, je pro překlad s nízkou chybovostí dobré jeho překladový program usměrnit; proto se ještě před statistickými výpočty přidává i fáze lingvistická, která zajišťuje to, že bude překladový program brát ohled na gramatiku a gramatické jevy daného jazyka. Jakmile tedy zkombinujeme statistické, lingvistické a další metody, vznikne tím strojový překlad, který nazýváme hybridní.

### 2.7.4 Hloubkově-syntaktický překlad

Překlad postavený na větném rozboru má za cíl zajistit lepší gramatický výstup. Algoritmus převádí větu na povrchovou a hloubkovou hladinu reprezentace, kterou můžeme vidět na Obrázek 6. Tímto způsobem vznikne strom větných členů a jednotlivé vztahy mezi nimi. Překlad do cílového jazyka se děje právě v hluboké reprezentaci, kdy se překládá strom na další strom. Paralelní slovník z toho důvodu neobsahuje všechny tvary slov, ale pouze základní tvary. O závěrečné skloňování či časování při vytváření cílové věty se starají samostatná komponenta systému.




*Obrázek 6: Povrchová hladina české a anglické věty. Každé slovo ve větě má ve větném rozboru svůj uzel, proto jsou si český a anglický stromy poměrně málo podobné. Šipky znázorňují slovní zarovnání.*

Systém s hlubokým překladem je sestaven z mnoha částí různého charakteru, pokud se na tento systém díváme z technického pohledu. O základní větný rozbor se starají statistické metody trénované na korpusech, které jsou rozšířené o morfologickou, syntaktickou a sémantickou anotaci. Při vytváření stromu na nový strom je možné aplikovat stabilní lingvistická pravidla, která upřesňují rozdíl mezi vstupní a výstupní větou.

### 2.7.5 Neuronové sítě

Použitím neuronových sítí se překlady výrazně zkvalitnily. Zatímco statistické metody se zaměřují na překlad po částech, u neuronových sítí je to naopak a překládají se kompletně celé věty. Díky překladu celých vět a ne pouze jejich částí je dosaženo lepších výsledků hlavně v kontextu.

Neuronová síť dokáže určit, do jaké míry jsou si podobná jednotlivá slova či fráze, proto překladový program založený na neuronové síti dokáže daleko lépe pracovat jak se slovy homonymními, tak i slovy, která se využívají zřídkakdy. Můžeme to uvést na následujícím příkladu, kdy se slovo „josta“ nevyskytuje v běžné mluvě moc často, ale neuronová síť si jej umí zařadit do stejné kategorie jako slova „rybíz“ či rybíz“, a poté s tímto slovem i pracovat. Neuronová síť také může dojít ke stejnému závěru i na základě toho, že se učí zároveň několik jazyků. Pokud bychom tedy měli více vět s použitím slova „josta“ ve vietnamštině než v němčině, odhadne neuronová síť na základě svých vědomostí z vietnamštiny, jak může o jostě hovořit v němčině.

Jako nejznámější a nejvíce rozšířený překladač, který používá neuronové sítě, můžeme uvést překladač od společnosti Google.

# 3 Rozhodovací stromy

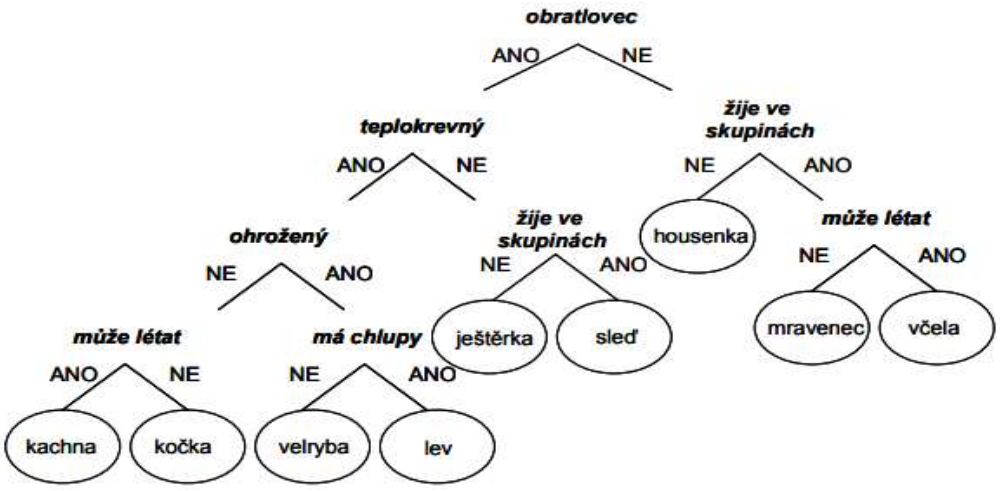
Rozhodovací stromy patří do oblasti klasifikace dat. Úkolem rozhodovacího stromu je popsat určité objekty za pomocí tzv. *příznakových vektorů*. Objektem můžeme chápat cokoliv, co lze popsat různými atributy. Atribut je rozlišovací rys a jeho hodnota může být jakákoli vlastnost objektu. Vlastnost objektu patří do kategorie *numerického* nebo *nominálního* charakteru. Zatímco numerický atribut má jakoukoli hodnotu z oboru reálných čísel $\mathbb{R}$, do nominálního atributu můžeme uložit pouze hodnoty z předem známého výčtu. Objekty jsou tedy klasifikovány na základě příznakových vektorů.

U rozhodovacích stromů je zřejmá podobnost se stromy v přírodě, proto byla převzatá tato terminologie, která dobře vystihuje podstatu algoritmu a je pro stromy běžná.

| | Teplokrevný | Může létat | Obratlovec | Ohrožený | Žije ve skupinách | Má chlupy |
|---|---|---|---|---|---|---|
| **Kočka** | ANO | NE | ANO | NE | NE | ANO |
| **Kachna** | ANO | ANO | ANO | NE | ANO | NE |
| **Sleď** | NE | NE | ANO | NE | ANO | NE |
| **Lev** | ANO | NE | ANO | ANO | ANO | ANO |
| **Ještěrka** | NE | NE | ANO | NE | NE | NE |
| **Velryba** | ANO | NE | ANO | ANO | ANO | NE |
| **Housenka** | NE | NE | NE | NE | NE | ANO |
| **Mravenec** | NE | NE | NE | NE | ANO | NE |
| **Včela** | NE | NE | NE | NE | NE | ANO |

*Tabulka 4: Kritéria pro rozdělení živočichů.*

Rozhodovací stromy se skládají z uzlů, hran a listů. Začáteční uzel se nazývá kořen, z kterého probíhá dělení do dalších uzlů. Pokud se uzly dál nerozkládají, nazýváme je listy nebo také terminální uzly. Stromy jsou děleny na binární a nebinární na základě toho, zda se rozvětvují na dvě nebo více větví.




*Obrázek 7: Ukázka rozhodovacího stromu. [11]*

Postupy, jakými rozhodovací stromy pracují, můžeme rozdělit podle toho, jaký charakter má zkoumaný problém na klasifikační a regresní. Co se týče klasifikační metody, zde je klíčová snaha rozřadit neznámý objekt do tříd, jejichž počet je striktně dán. Zde vycházíme z dat, která už byla předem naměřená. Určíme kritéria důležitá pro jednotlivé třídy závislé proměnné a na jejich podkladu vytvoříme model popisující zkoumaný problém tak, že je možné zařadit do správné třídy nové parametry, které jsou klíčové pro jednotlivé kategorie závisle proměnné, a na základě těchto parametrů se vytvoří model, který popisuje určitý problém tak, aby byl schopen zařadit do správné kategorie nová neznámá data. Druhý postup, regresní, je takový, kdy má závislá proměnná kvantitativní povahu. Použitím regresního stromu můžeme odhadnout, jakou hodnotu bude mít určité číselné kritérium. Výsledkem použití regresního stromu je rovnice, která má odhadnuté regresní veličiny a kterou lze predikovat hodnoty závislé proměnné. Můžeme tedy souhrnně říci, že rozdíl mezi klasifikačním a regresním rozhodovacím stromem je v druhu závislé proměnné. [11]

- **Klasifikační rozhodovací strom** – vytváříme závislost **kategoriální** závislé proměnné na jedné či více nezávislých proměnných.

- **Regresní rozhodovací strom** – vytváříme závislost **spojité** závislé proměnné na jedné či více nezávislých proměnných.

## 3.1 Algoritmy vytvářející rozhodovací stromy

Cílem této podkapitoly není podat vyčerpávající seznam algoritmů pro sestavení rozhodovacích stromů, ale představit dva základní zástupce algoritmů pro toto sestavení.

### 3.1.1 ID3

Algoritmus ID3 při tvorbě rozhodovacího stromu vychází z principu, který je nazýván jako *Occamova břitva*. Tento princip říká, že jakmile existuje více možností, jak lze objasnit určitý jev, je lepší upřednostnit tu nejméně obtížnou možnost. Co se týče rozhodovacích stromů, zde se za nejméně komplikovanou cestu považuje zároveň i ta nejkratší, tedy cesta od kořene k listu. Co nejmenší strom můžeme vytvořit tím, že použijeme takové atributy, u kterých existuje velký předpoklad, že ukončí růst tohoto stromu co nejrychleji.

### 3.1.2 C4.5

Algoritmus C4.5 byl vytvořen po algoritmu ID3. Základ, na kterém pracuje, je stejný jako u algoritmu ID3. Je tam ovšem rozdíl v tom, že místo informačního zisku pracuje s tzv. poměrným informačním ziskem. Na rozdíl od informačního zisku však řeší problém, který spočívá v tom, že pokud je hodnota určitého atributu jednoznačná (například rodné číslo), pak se tento atribut použije v kořeni stromu, tím pádem do každého podstromu spadne pouze jeden příklad a růst se zastaví. Poměrný informační zisk toto bere v potaz a zohledňuje i počet těchto atributových hodnot. Vedle této nesporné výhody má oproti algoritmu ID3 také výhody spočívající například v tom, že se dokáže učit z příkladů, ve kterých chybí hodnoty atributů nebo osekání výsledného stromu atp.

# 4 Vlastní řešení

Pokud bychom přistoupili ke korekci slov jako k jednojazyčnému překladu, pak bychom měli chybnou větu považovat za zdrojovou. Přetransformováním této věty bychom měli dostat větu plynulou, gramaticky a pravopisně správně, která by si však zároveň zachovala svůj informativní obsah.

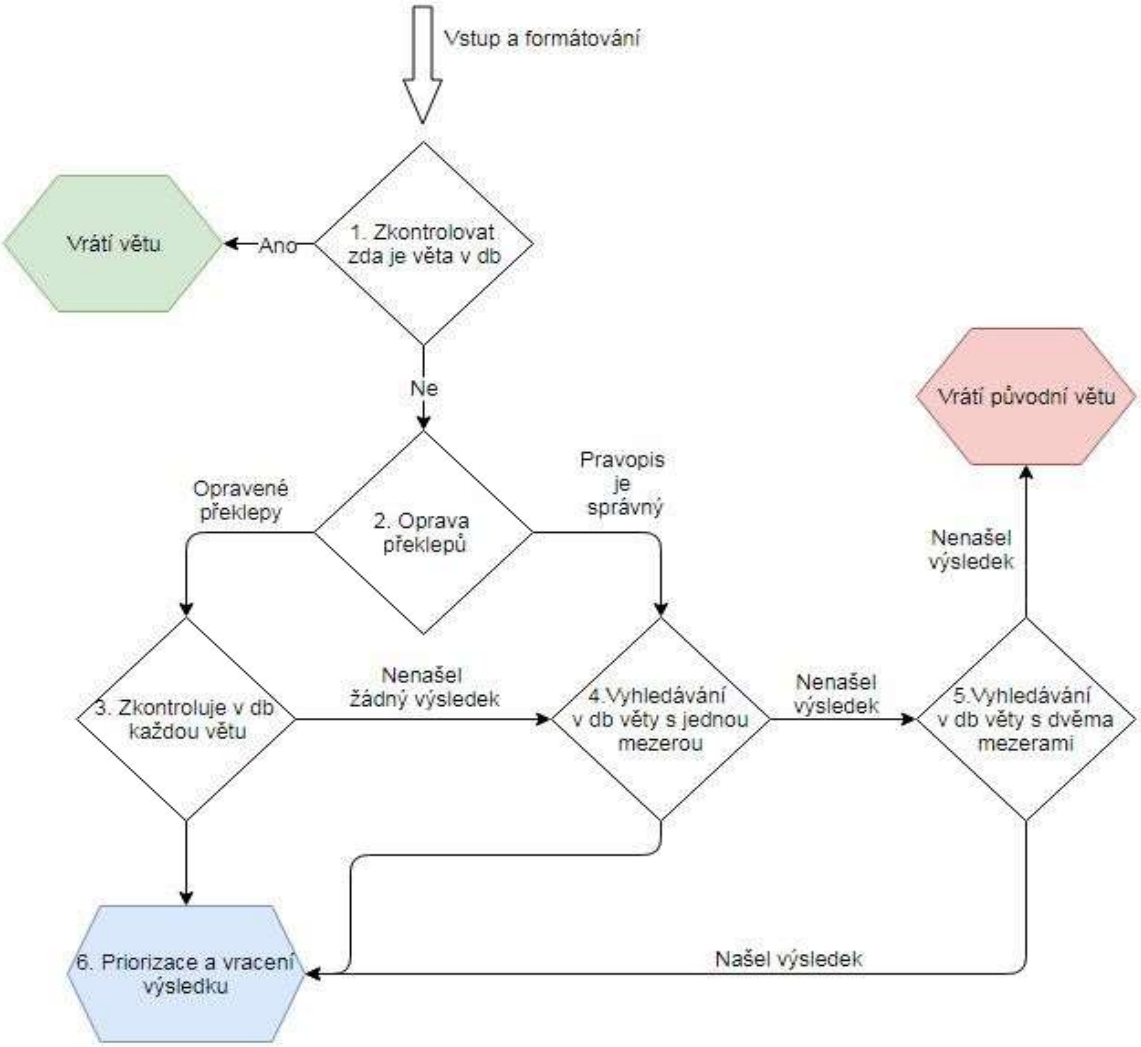
Důvodem vzniku tohoto algoritmu je neformalizovatelnost přirozeného jazyka. [11] Díky tomu nemůžeme uplatnit sadu několika pravidel přirozeného jazyka a očekávat vysokoprocentní úspěšnost; zvláště pak ne u českého jazyka, který má volnou větnou skladbu. Proto byl vybrán jako řešení korekce textu rozhodovací strom.

Tento model se však nesnaží opravit vstupní text ve smyslu implementace pravidel pravopisu a gramatiky. Model, jehož vývojový diagram je znázorněn na Obrázek 8, se snaží predikovat gramatickou a pravopisnou chybu porovnáním jednotlivých slov nebo vět s těmi, u kterých již proběhla lidská korekce a které jsou uloženy v databázi (více o databázi v kapitole 4.7). Porovnáním vstupního textu s již existujícími texty v databázi jsme schopni určit, zda je vstupní text v korektním tvaru či nikoli, a případně tento text dál upravit do takových podob, aby nebyl příliš odlišný od originálního textu (resp. vstupního textu), ale zároveň se shodoval s textem, který již existuje v databázi. Algoritmus může vrátit více výsledků; v tom případě se přistupuje k vyhodnocování toho, která věta se nejvíce podobá originální, ať už podle kontextu celého textu nebo metodami pro zjištění nejpodobnější věty s originální větou.

Algoritmus byl vytvářen pro korekci textu v jazyce českém a také v jazyce českém byl testován. Celý algoritmus by bylo potřeba přepracovat, pakliže bychom zvolili jiný jazyk, zvlášť takový, který má pevnou větnou skladbu.

Z popisu funkčnosti algoritmu je patrné, že algoritmus je omezen pouze na korekci těch vět, které zná a které jsou uloženy v korpusu.

Dále v textu jsou podrobně vysvětleny jednotlivé kroky algoritmu, které jsou také vyobrazeny na Obrázek 8. [12]




*Obrázek 8: Vývojový diagram algoritmu automatické korekce textu pro jazyk český.*

## 4.1 Vstup a formátování

Před spuštěním samotného algoritmu je třeba, aby byl celý text určitým způsobem formátován a rozdělen na jednotlivé věty. Každá věta textu je následně zpracovávána samostatně.

### 4.1.1 Rozložení textu na jednotlivé věty

Text, který uživatel vloží do aplikace pro korekci, je třeba rozdělit na jednotlivé věty. Tyto věty následně projdou algoritmem pro určení jejich správnosti, případně chybovosti, a dojde k jejich automatické korekci. Text bez hlubších pravidel je rozdělen pomocí znaků „.,!,?“ (tečka, vykřičník, otazník) s podmínkou, že následující znak je velké písmeno. Pozorný čtenář by mohl namítnout, že je spousta situací, kdy uvedené pravidlo neplatí.

Pro příklad uvedu text:

*Vývoj v dějinách českého státu procházel významnými proměnami. 26. září 1212 vydal v Basileji Fridrich II. jako odměnu za podporu Přemyslu Otakarovi I. Zlatou bulu sicilskou.* Další příklad může vypadat následovně: *Maminka přišla domů. „Děti, pojďte k večeři.“*

Tyto příklady poukazují na nedostatečnost rozdělení textu pouze podle pravidla ukončujícího znaku (tečka, vykřičník, otazník) a následujícím velkým písmenem. Rozdělení textu na jednotlivé věty podle sémantických pravidel je nad rámec této bakalářské práce.

### 4.1.2 Formátování vět

Každá věta má několik způsobů naformátování. V této kapitole jsou ukázány a vysvětleny všechny formáty, které jsou nezbytné pro funkčnost algoritmu.

Základní formát pro každou větu je odstranění názvů, jmen a čísel. Tyto prvky jsou nahrazeny znakem „#“. Zbytek věty je zachován v originálním tvaru.

Formát úrovně 1: Navíc dolar dost posiluje proti euru, # komentoval.

Dalším formátem je odstranění mezer ve větě.

Formát úrovně 2: Navícdolardostposilujeprotieuru,#komentoval.

Poslední formát je vytvořen tak, aby v něm nebyly čárky ve větě, dvojtečky či znaménko na konci věty. Také odstraní všechna velká písmena, která jsou nahrazena ekvivalentem v malé verzi.

Formát úrovně 3: navícdolardostposilujeprotieuru#komentoval

## 4.2 Pravopisné chyby

Prvním důležitým krokem algoritmu je zjištění, jestli věta, na kterou je aplikován algoritmus, je správně, či nikoliv. Pro tento účel nám poslouží vyhledání všech naformátovaných verzí věty s větami korpusu. Pro vyhledávání je použita 3. úroveň formátování, které může vrátit více výsledku. Pokud vyhledávání vrátí více výsledků, pak je na řadě základní priorizace.

Cyklicky je porovnávána originální věta ve všech formátech s větou, která se nachází v korpusu. Díky porovnávání je vytvořen řetězec, jenž funguje jako klíč. Věta s řetězcem nejvyšší hodnoty „vyhrává“ a je vrácena se statusem „správná“.

a) Porovnání formátu 3
b) Porovnání formátu 2
c) Věta, která se nachází v databázi s největším počtem

Pro příklad uvedu následující klíče: *1_1_15* a *1_0_11*. V prvním případě má klíč shodu v bodě *A* a také shodu v bodě *B* a má celkem 15 opakování v databázi; ve druhém případě má klíč shodu v bodě *A*, ale už se neshoduje v bodě *B* a počet opakování v databázi je celkem 11. *Z* těchto dvou příkladu by byla vybrána věta s klíčem *1_1_15*.

Pokud vyhledávání věty 3. úrovně formátování nevrátí žádný výsledek, pak se jedná o potenciálně špatnou větu. A je třeba aplikovat další kroky algoritmu.

## 4.3 Algoritmus pro opravu překlepů

V této fázi algoritmu pracujeme s větou, která je potenciálně špatně; musíme zjistit, o kterou chybu se pravděpodobně jedná. Proto si jako první krok zjistíme, jestli každé slovo obsažené v aktuální větě existuje ve slovníku. Více o slovníku je probíráno v kapitole 4.7.

Pokud všechna slova kontrolované věty jsou obsažena ve slovníku, pravděpodobně jde o gramatickou chybu a algoritmus pokračuje postupem, který je uveden v kapitole 4.4. Pokud u slov kontrolované věty dojde k neshodě se slovníkem, pak jde pravděpodobně o překlep.

Pro každý výraz, který nebyl nalezen ve slovníku, jsou generovány náhrady. Generování náhrad probíhá dvěma způsoby.

V prvním případě se provede postupná záměna všech písmen z abecedy a mezery za každé písmeno daného výrazu, pro který se náhrady vytvářejí (pozn. písmeno ch není v algoritmu chápáno jako jedno písmeno, ale jako dvě písmena, tzn. Levensteinova délka vrací pro písmeno ch hodnotu 2, pokud budeme porovnávat prázdný znak s písmenem ch). Každý takto vygenerovaný řetězec znaků porovnáváme se slovníkem slov, a pokud je nalezena shoda, tak je daný řetězec uložen do listu jako náhrada za původní výraz. Výsledek každé náhrady je podle Levensteinova algoritmu maximálně 2. Pro příklad uvedu pár výsledků pro výraz „diler“ s Levensteinovou vzdáleností 1 : dialer, doler a s Levensteinovou vzdáleností 2: dle, dcer, diet, dslr, čile, kile, mile, pile, sile, tile, vile, dále, déle, dole, důle, dime, boiler, mailer, daimler, dealer, driver, drilem, dialerů, dialery, hitler, dilema, celer, cifer, citer, filtr, filek, files, filet, kilem, liber, liter, lilek, mixer, milec, niger, niter, silur, silem, šifer, vilek, žilek, dalar, dalek, dálek, délek, dělem, dělen, dílec, dílek, dílem, dílen, dober, dožer, dolar, dolec, dolem, dolet, důlek, džber, dinem, divem.

V druhém případě je postupováno pomocí permutace a jsou využita všechna písmena daného výrazu, s kterými jsou vytvořeny všechny možné variace, které jsou následně porovnávány se slovníkem. Ty výrazy, které najdou shodu ve slovníku, jsou připojeny k seznamu náhrad z prvního případu.

Každá náhrada z takového seznamu je následně zaměněna za původní výraz, v našem případě za výraz „diler“. Tímto vytvoříme pro každou náhradu potenciálně správnou větu, která má každé slovo pravopisně správně, resp. to znamená, že každé slovo, které se vyskytuje v takto vytvořených větách, nalezneme ve slovníku. Každá vytvořená věta pomocí těchto náhrad je následně testována, jestli již existuje v databázi. Pokud nalezneme jeden nebo více výsledků, pak tyto výsledky pokračují k priorizaci nejpřesnější věty. Tento krok je podrobně vysvětlen v kapitole 4.6. Pokud žádná z těchto vět není nalezena v databázi textů, pak můžeme odhadnout, že chyba není pravděpodobně v pravopise, ale v gramatice. Se všemi vygenerovanými větami bude algoritmus pokračovat v následujícím kroku.

## 4.4 Vyhledávání s jednou mezerou

V tomto kroku algoritmu se nacházejí pouze věty, které byly v předešlých krocích vyhodnoceny jako pravopisně správné; tedy každé slovo, které tyto věty obsahují, se zároveň nacházejí ve slovníku. Ačkoliv každé slovo bylo nalezeno ve slovníku, tak celá věta nebyla nalezena v databázi vět, nad kterými už korekce proběhla (více v kapitole 4.7). Proto je velice pravděpodobné, že ve větě nebo větách v tomto kroku nějaké slovo chybí nebo přebývá.

Pro každou větu v tomto kroku je tak třeba vygenerovat její repliku s vždy chybějícím jedním slovem, které nemusí zapadat do kontextu. Tímto krokem nalezneme slova, která jsou přebytečná anebo do daného kontextu nepatří vůbec. Pro příklad je zde uvedena věta: „Americký mince opět čelí mezinárodní kritice.“ Ačkoliv je tato věta po pravopisné stránce napsaná bezchybně, významově již smysl nedává (více v kapitole 2.4). Z toho důvodu vygenerujeme repliky vět (v ukázkovém případě symbol *** nahrazuje chybějící slovo):

| | | | | | | |
|---|---|---|---|---|---|---|
| *1.* | *** | *mince* | *opět* | *čelí* | *mezinárodní* | *kritice.* |
| *2.* | *americký* | *** | *opět* | *čelí* | *mezinárodní* | *kritice.* |
| *3.* | *americký* | *mince* | *** | *čelí* | *mezinárodní* | *kritice.* |
| *4.* | *americký* | *mince* | *opět* | *** | *mezinárodní* | *kritice.* |
| *5.* | *americký* | *mince* | *opět* | *čelí* | *** | *kritice.* |
| *6.* | *americký* | *mince* | *opět* | *čelí* | *mezinárodní* | ***. |

*Tabulka 2: Tabulka s vynechaným jedním slovem.*

Po vygenerování vět s mezerou už jsme schopni nalézt takovou větu, při níž autor mohl udělat překlep, který je pravopisně správně. Pořád však může nastat situace, kdy autor mohl vynechat slovo, které je pro správnost věty taktéž důležité. Potřebujeme tak dát mezery (opět symbol ***) mezi každé dvě slova a také na začátek a na konec věty, abychom tak přišli na slovo, které autor mohl vynechat.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *7.* | *** | *americký* | *mince* | *opět* | *čelí* | *mezinárodní* | *kritice.* |
| *8.* | *americký* | *** | *mince* | *opět* | *čelí* | *mezinárodní* | *kritice.* |
| *9.* | *americký* | *mince* | *** | *opět* | *čelí* | *mezinárodní* | *kritice.* |
| *10.* | *americký* | *mince* | *opět* | *** | *čelí* | *mezinárodní* | *kritice.* |
| *11.* | *americký* | *mince* | *opět* | *čelí* | *** | *mezinárodní* | *kritice.* |
| *12.* | *americký* | *mince* | *opět* | *čelí* | *mezinárodní* | *** | *kritice.* |
| *13.* | *americký* | *mince* | *opět* | *čelí* | *mezinárodní* | *kritice* | ***. |

*Tabulka 3: Tabulka zobrazující věty mezerou pro nové slovo.*

U každé takto vygenerované věty je důležité znát pořadí každého slova ve větě a také jeho číselný identifikátor, který nalezneme ve slovníku. Pro příklad je zde uvedena generovaná věta: *americký *** opět čelí mezinárodní kritice.*

| **Slovo** | **Pořadí** | **Identifikátor slovníku** |
|---|---|---|
| americký | 1 | 5287 |
| opět | 3 | 436197 |
| čelí | 4 | 899452 |
| mezinárodní | 5 | 210048 |
| kritice | 6 | 172289 |

*Tabulka 4: Tabulka ukazující pořadí slov ve větě a jejich identifikační číslo ve slovníku*

Následně jsou v databázi vyhledány věty, kde jsou stejná slova ve stejném pořadí, čímž bychom měli získat slovo chybějící. V tomto případě algoritmus vyhledá větu: *„Americký dolar opět čelí mezinárodní kritice,“* kde chybějící slovo bylo dolar.

Tento krok je však výpočetně velice náročný, jelikož takto vygenerovaných vět mohou být tisíce, obzvlášť v případě, že se v původní větě nacházela pravopisná chyba (viz kapitola generování náhrad v pravopisu). Z tohoto důvodu byla v tomto kroku vytvořena v databázi dočasná tabulka `#tempwordsinsentences`. Při testování bylo zjištěno, že cyklické nahrávání dat klasickým způsobem pomocí příkazu „`INSERT INTO #tempwordsinsentences`“ je časově velice neefektivní, proto v tomto bodě bylo použito pro nahrávání dat do dočasné tabulky `#tempwordsinsentences` hromadné kopírování pomocí třídy *SqlBulkCopy*.

```
StringBuilder tmpTable = new StringBuilder();

tmpTable.AppendLine("CREATE TABLE #tempwordsinsentences");
tmpTable.AppendLine("  ( ");
tmpTable.AppendLine("     [SentenceId]  INT NOT NULL, ");
tmpTable.AppendLine("     [WordId]      INT NOT NULL, ");
tmpTable.AppendLine("     [Rank]      INT NOT NULL ");
tmpTable.AppendLine("  ); ");

DataTable table = new DataTable();
table.Columns.Add(new DataColumn("SentenceId", typeof(int)));
table.Columns.Add(new DataColumn("WordId", typeof(int)));
table.Columns.Add(new DataColumn("Rank", typeof(int)));

int SentenceId = 1;

for (int i = 0; i < wordsIds.Length; i++)
{
    table.Rows.Add(SentenceId, wordsIds[x], (x + 1));
    DataRow row = table.NewRow();
    i += 1;
}

SqlCommand cmd = new SqlCommand(tmpTable.ToString(), connection);
cmd.ExecuteNonQuery();

SqlBulkCopy bulk = new SqlBulkCopy(connection);
bulk.DestinationTableName = "#tempwordsinsentences";
bulk.WriteToServer(table);
```

*Obrázek 9: Ukázka kódu použití třídy SqlBulkCopy. [13]*

Vět, kterých algoritmus nalezl v tomto pořadí, může být mnoho; proto pokud algoritmus najde více než jednu větu, následně aplikuje tzv. metodu priorizace (více v kapitole 4.6). Pokud v tomto kroku algoritmus v databázi nenalezne žádný výsledek, pokračuje vstup vět do tohoto kroku dalším krokem.

## 4.5 Vyhledávání s dvěma mezerami

Tento krok je velice podobný s krokem předchozím, který je podrobně popsán v kapitole 4.4 s tím rozdílem, že ve vygenerovaných větách není pouhá jedna mezera pro náhradní slovo, ale mezery jsou dvě.

$$C = \frac{L!}{2! \times (L-2)!} + L^2 + \frac{(L+2)!}{2! \times [(L+2)-2]!}$$

*Rovnice 2: Počet vygenerovaných vět.*

Kdy *C* je počet výsledných vět a *L* je počet slov věty. Pro příklad uvedu větu: „*Koupil jsem auto.*“

| | | | |
|---|---|---|---|
| 1 | **** *** koupil jsem auto.* | 12 | **** jsem *** auto.* |
| 2 | **** koupil *** jsem auto.* | 13 | **** jsem auto ***.* |
| 3 | **** koupil jsem *** auto.* | 14 | **** koupil *** auto.* |
| 4 | **** koupil jsem auto ***.* | 15 | *koupil *** *** auto.* |
| 5 | *koupil *** *** jsem auto.* | 16 | *koupil *** auto ***.* |
| 6 | *koupil *** jsem *** auto.* | 17 | **** koupil jsem ***.* |
| 7 | *koupil *** jsem auto ***.* | 18 | *koupil *** jsem ***.* |
| 8 | *koupil jsem *** *** auto.* | 19 | *koupil jsem *** ***.* |
| 9 | *koupil jsem *** auto ***.* | 20 | **** *** auto.* |
| 10 | *koupil jsem auto *** ***.* | 21 | **** jsem ***.* |
| 11 | **** *** jsem auto.* | 22 | *koupil *** ***.* |

*Tabulka 5: Tabulka zobrazující hledání dvou náhrad ve větě.*

Jako v předešlém kroku je porovnávána každá vygenerovaná věta s větami v databázi, kdy jsou vyhledávány věty v databázi, které mají ve stejném pořadí slova jako věty vygenerované. Pokud výsledek hledání je počet vět větší než jedna, pak výsledné věty pokračují krokem priorizace vět, viz kapitola 4.6. Pokud výsledné hledání nevrátí větu žádnou, pak algoritmus automatické korekce selhal a je vrácena původní věta.

## 4.6 Priorizace vět

Aktuální programy, které aplikují korekci textu, pracují ve dvou krocích. V prvním kroku algoritmus analyzuje text a detekuje chybu. V druhém kroku algoritmus vyhodnocuje možnosti opravy, které následně nabídne autorovi, který se následně rozhodne, zda využije nabídnuté opravy. U automatické korekce je tato možnost ponechána čistě na samotném algoritmu, který musí vyhodnotit, která z oprav je nejvhodnější. Na základě této skutečnosti musí být vytvořen způsob, který by dokázal vybrat vhodnou větu, pokud by nastala situace, ve které by bylo možných více variant. Tento způsob tkví v posloupnosti několika kroků, které byly vybrány na základě testování a empirického bádání. V následujících bodech jsou popsány jednotlivé kroky, které jsou nezbytné pro správné ohodnocení vygenerovaných vět. Na základě hodnocení jednotlivých vět je vybrána nejvhodnější věta, která by mohla nahradit větu originální (chybnou).

1. Každé vygenerované větě je přiřazeno jako základní hodnocení 999, ze kterého je odečtena Levensteinova délka původní a generované věty. Tímto krokem získáme větu, která je nejpodobnější původní větě.
2. Druhá část klíče se skládá z počtu nalezených náhrad v celém článku, který byl do aplikace zadán. Tímto můžeme nalézt slovo, které se nejčastěji nachází v textu; tímto krokem se snažíme odhadnout, zda dané slovo zapadá do kontextu.
3. Posledním krokem klíče je využití informace týkající se počtu výskytu vygenerované věty v databázi. Je to z toho důvodu, že frekvence užití této věty je vyšší a statisticky je tedy pravděpodobnější její výskyt.

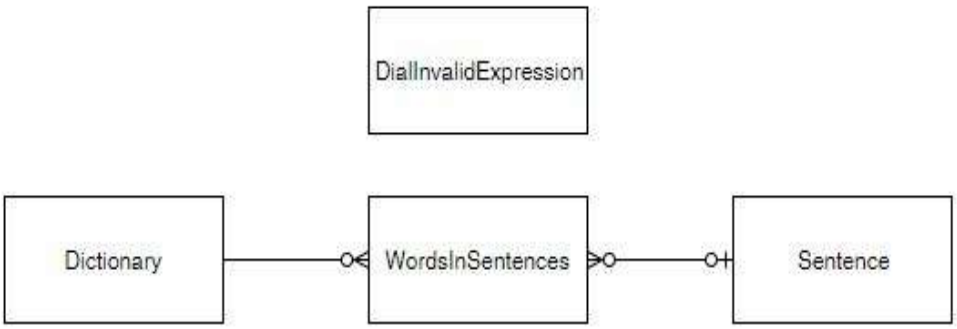
## 4.7 Databáze

Jak bylo zmíněno na začátku této práce, rozdíl mezi automatickou korekcí a poloviční korekcí je v odpovědnosti. U poloviční korekce je odpovědný za výsledek vždy člověk, u automatické korekce je za výsledek odpovědný automat. Aby algoritmus mohl vyhodnocovat správnost výsledků, je třeba, aby tyto výsledky mohl porovnávat s již existujícími větami, které v minulosti prošly korekcí. Z tohoto důvodu byly v této bakalářské práci využity články ze serveru novinky.cz, které byly publikovány v posledních deseti letech pro vytvoření vlastního korpusu textů. Tyto texty byly staženy vytvořeným skriptem, který zároveň data zpracovával do podoby, kterou algoritmus pro správné fungování vyžaduje. Textový korpus obsahuje 287 000 jednotlivých vět z 50 000 článku, které byly staženy a zpracovány. Výhodou článku ze serveru novinky.cz je jejich tematická různorodost, jako je např. kultura, finance, sport, politika atd. [14]

U každé věty byly použity tři druhy formátů, které jsou vysvětleny v kapitole 4.1.2. U jednotlivých vět je velice důležité, aby měly také informaci, v jakém pořadí se nacházejí jednotlivá slova. O pořadí jednotlivých slov ve větách dále v kapitole 4.4.

Jako další neméně důležitou součástí dat je slovník. Abychom mohli o nějakém slově říct, jestli je nebo není ve správném pravopisném tvaru, musíme tento tvar také znát. Z tohoto důvodu algoritmus využívá slovník českých slov, která by v tomto slovníku měla být ve všech tvarech a čítá tak 925 797 záznamů. [15]

V této bakalářské práci je využita MS-SQL databáze jako datové úložiště, do kterého je uložena datová struktura, která je vyobrazena ER diagramem na Obrázek 10.




*Obrázek 10: ER diagram.*

### 4.7.1 Popis datových tabulek

V této podkapitole budou stručně popsány jednotlivé tabulky databáze, které jsou nezbytnou součástí algoritmu. Na Obrázek 10 můžeme vidět ER diagram struktury databáze.

- **Dictionary**

**Obsah**: Tabulka *Dictionary* obsahuje korpus všech slov a jejich tvarů, která se v českém jazyce vyskytují. Ke každému slovu je přiřazen také unikátní identifikátor.

- **Sentences**

**Obsah**: Tabulka *Senteces* obsahuje všechny věty, které byly získány ze serveru www.novinky.cz. Věty uloženy v této tabulce jsou po lidské korekci, a mohou tedy být využity jako vzor pro správné fungování algoritmu.

- **WordsInSentences**

**Obsah**: Tabulka *WordsInSentences* je vazební tabulkou mezi tabulkami *Sentences* a *Dictionary*. Tato tabulka obsahuje informaci o pozici slov ve větě, tzn. jaké slovo je v jaké pozici v jaké větě.

- **DialInvalidExpression**

**Obsah**: Tabulka *DialInvalidExpression* obsahuje zkratky a tituly. Po těchto výrazech následuje vždy tečka, která by více komplikovala nalezení konce věty. Na základě komplikovanosti nalezení konce věty bylo nutné sbírat takové výrazy, po kterých se píše tečka, která však neznamená ukončení věty. Tyto výrazy jsou na začátku algoritmu odstraněny a na konci textu vráceny zpět.

# 5 Vyhodnocení výkonnosti a efektivity

Hlavní parametry výkonnosti vytvořeného algoritmu jsou rychlost výpočtu korekce daného textu a efektivita. Abychom mohli vyhodnotit efektivitu korekce algoritmu a zároveň rychlost zpracování, je potřeba vystavit algoritmus několika testům.

## 5.1 Vyhodnocení efektivity

V této podkapitole nás bude zajímat, do jaké míry je algoritmus schopný ještě opravit text a kdy už je vstupní text v tak špatném stavu, že už si s ním algoritmus nedokáže poradit. Testovat budeme na náhodně vybrané větě z databáze, u které uměle vytvoříme několik chyb, a budeme zkoumat, kdy už si algoritmus není schopen s textem poradit. Testovat budeme na náhodně vybraných větách z databáze.

*Roshodl o tom fýkonný výbor americké filmové akademie. Proti reformě sei přesto tehdy postavil lidovecký poslanec. Jsou diskriminováni při dědění adarování. Nechceme bít zneužívány k likvidačním a bomocným pracím. Hasiči to zjistí ažpo příjezdu na místo. Podle něj měly oba stejnou šanci se vyjádřit. Aluminiové disky natestované vozy instalovala automobilka sama. Závažnější problém vžak přišel při bočným nárazu. Bezvládného děla na dně bazénu si všimla jedna z plafkyň. Bes ní bych asi nic nenapsal. Úspěch té knýšky mě překvapil, pořád ho úplně nechápu. Na místě bylo víc polycystů a novinářů než protestujících. Alespoňto tvrdý někteří ochránci přírody. Učitelé je navýc buď vůbec nesnají, nebo nepoužívají. Kurikulární revorma nezkončila, ta nejnáročnější část je teprve před námi. Zavoláným na něj mohou přijít o mnoho peněz. Prodražit semůže až zpětné volání. Běžně šlo o zto devadesát kilometrů. Sněšku podobný vítr naposledy provětral téměř přesně před rokem. Kriminalisté na místě našli ikladivo, které vrah nejspíš použil. Na kandidátku tehdi napsal jména lidí, kteří o tom vůpec nevěděli. Odmítl zpekulace médií, že požár uvěsnil v budově několik lidí. Zejména v horskích a podhorských oblastech porasil několik stromů. Při cvyčení se však ukázaly jako kvalitní. Stroje vyráběné v malých nebo v seškrtaných seriích se prodražují jak výrobně, tak při údržbě, škrty počtu dodaných strojů nesnižují náklady na vývoj a přípravu výroby. Novinky srovnávaly nejfrekventovanější trazu v republice co do ceny a doby cesty nejrůznějšými dopravními prostředky. Vžichni dopravci však nabízejí nejrůznější slevy, pokud má cestující například nějakou předem zakoupenou průkazku a podobně.*

*Text 1: Text obsahujíc uměle vytvořené chyby.*

*Rozhodl o tom výkonný výbor americké filmové akademie. Proti reformě se i přesto tehdy postavil lidovecký poslanec. Jsou diskriminováni při dědění a darování. Nechceme být zneužíváni k likvidačním a pomocným pracím. Hasiči to zjistí až po příjezdu na místo. Podle něj měli oba stejnou šanci se vyjádřit. Aluminiové disky na testované vozy instalovala automobilka sama. Závažnější problém však přišel při bočním nárazu. Bezvládného těla na dně bazénu si všimla jedna z plavkyň. Bez ní bych asi nic nenapsal. Úspěch té knížky mě překvapil, pořád ho úplně nechápu. Na místě bylo víc policistů a novinářů než protestujících. Alespoň to tvrdí někteří ochránci přírody. Učitelé je navíc buď vůbec neznají, nebo nepoužívají. Kurikulární reforma neskončila, ta nejnáročnější část je teprve před námi. Zavoláním na něj mohou přijít o mnoho peněz. Prodražit se může až zpětné volání. Běžně šlo o sto devadesát kilometrů. Sněžku podobný vítr naposledy provětral téměř přesně před rokem. Kriminalisté na místě našli i kladivo, které vrah nejspíš použil. Na kandidátku tehdy napsal jména lidí, kteří o tom vůbec nevěděli. Odmítl spekulace médií, že požár uvěznil v budově několik lidí. Zejména v horských a podhorských oblastech*

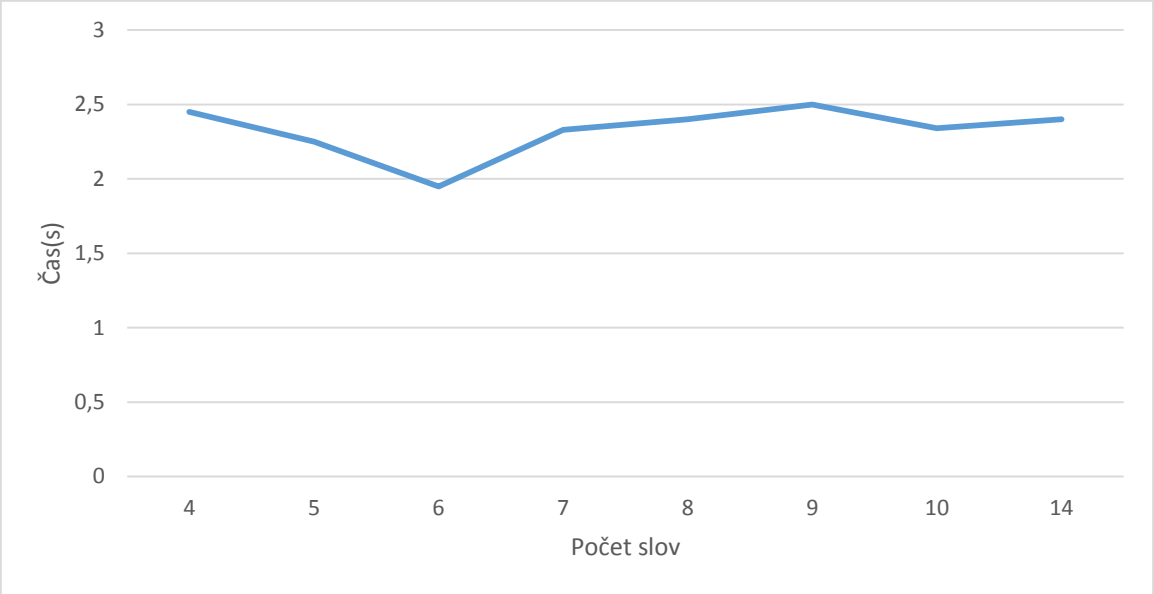
*porazil několik stromů. Při cvičení se však ukázaly jako kvalitní. Stroje vyráběné v malých nebo v seškrtaných sériích se prodražují jak výrobně, tak při údržbě, škrty počtu dodaných strojů nesnižují náklady na vývoj a přípravu výroby. Novinky srovnávaly nejfrekventovanější trasu v republice co do ceny a doby cesty nejrůznějšími dopravními prostředky. Všichni dopravci však nabízejí nejrůznější slevy, pokud má cestující například nějakou předem zakoupenou průkazku a podobně.*

*Text 2: Výsledek algoritmu je kompletně opravený text.*

## 5.2 Vyhodnocení rychlosti

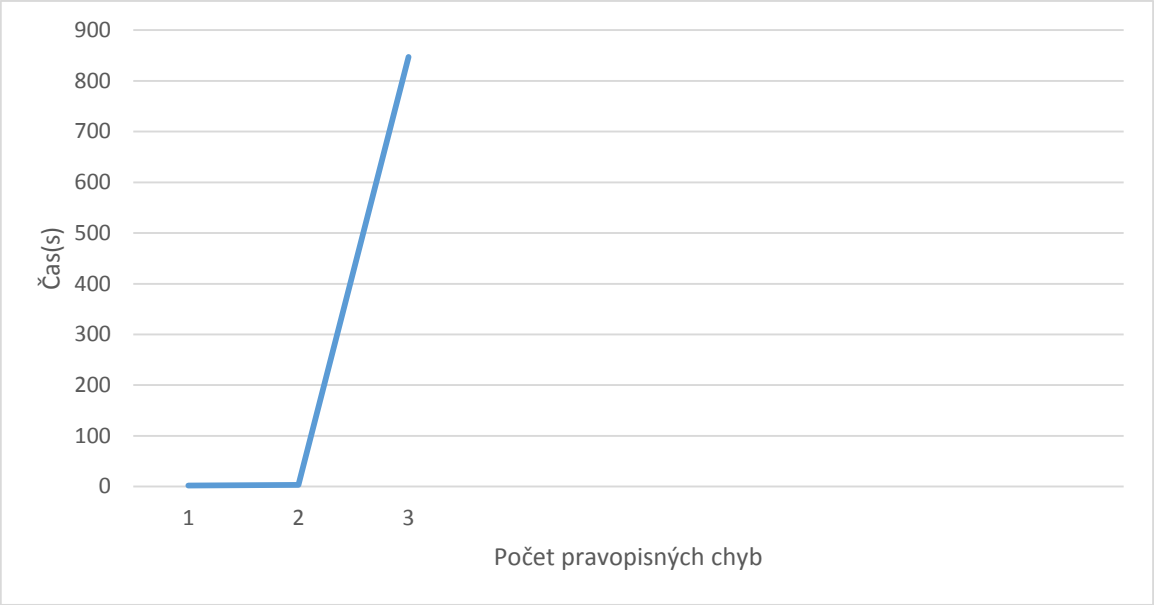
Pro finální měření rychlosti algoritmu byl použít stejný počítač, který byl použit pro vývoj. Jedná se o přenosný notebook s následující konfigurací operační systém: Windows 7 64 bit, CPU: Intel Core i3-2310 M 2,1 GHz, Paměť RAM: 6,0 GB. Vyvinutý systém používá pro korekci velké množství dat. Během výpočtu se může dosahovat v mezi tabulkách až ke stovkám milionů záznamů. Algoritmus byl testován na jednotlivých větách z důvodu velké časové náročnosti.

Výkonnost byla měřena na třech úrovních. První měření zkoumá časovou náročnost vět s navyšujícím se počtem slov o jedné pravopisné chybě. V druhém případě se jednalo o stejnou větu s deseti slovy a navyšujícím se počtem pravopisných chyb, měřena byla časová náročnost. V třetím případě byla testována rychlost opravy gramatické chyby s přibývajícím počtem slov ve větě.



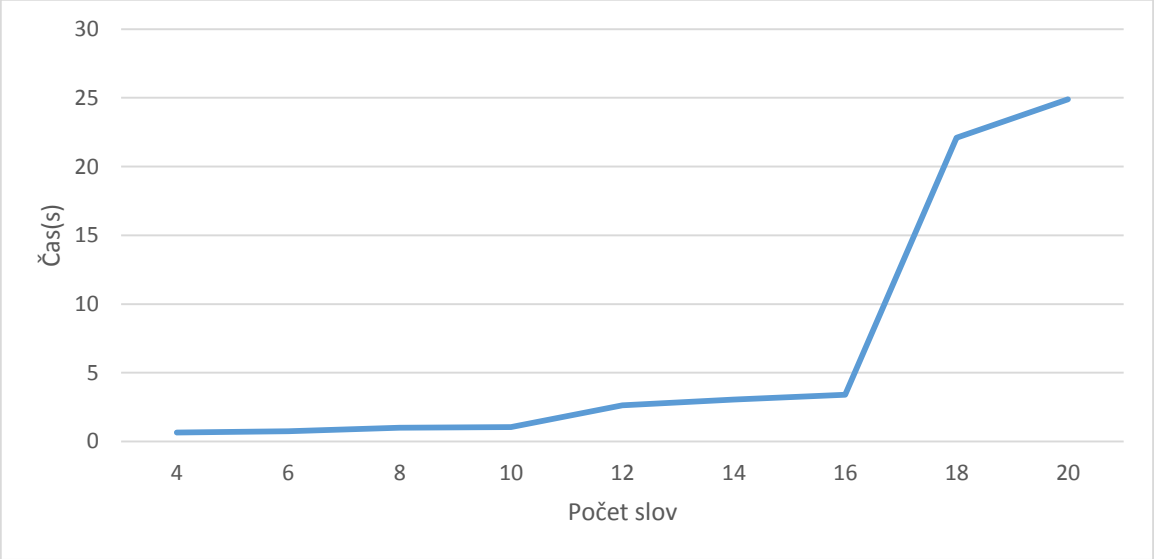

*Graf 1 Zvyšující se počet slov ve větě o jedné pravopisné chybě.*

Z grafu 1 je zřejmé, že čas výpočtu není ovlivněn množstvím slov ve větě, pokud věta obsahuje pouze jednu pravopisnou chybu. Z tohoto důvodu bylo potřeba otestovat výkonnost algoritmu rostoucím počtem pravopisných chyb nad konstantním počtem slov ve větě.




*Graf 2 Zvyšující se počet pravopisných chyb.*

Z grafu 2 vyplývá, že algoritmus je schopen opravit větu s dvěma pravopisnými chybami za přijatelný čas. Při navýšení počtu slov se zahltí paměť RAM a časová náročnost algoritmu pro opravu věty přestává být v přijatelných mezích. Pro rychlejší funkčnost by byla potřeba navýšit paměť RAM, která by snížila potřebný čas pro výpočet s narůstajícími pravopisnými chybami.




*Graf 3 Zvyšující se počet slov ve větě o jedné gramatické chybě.*

Graf 3 má stoupající tendenci lineárního charakteru do okamžiku, kdy se ve větě nachází maximálně 16 slov. Jakmile je tato hranice překročena, povaha nárůstu grafu se výrazně navyšuje a naměřený čas, za který je věta opravena, skokově narůstá.

# 6 Závěr

Cílem této práce bylo vytvořit systém pro korekturu textu. Zároveň jedním z úkolů bylo seznámit čtenáře s metodami, které řeší korekturu textu. Po seznámení se s metodami, které byly vyhodnoceny jako nevyužitelné, následoval návrh vlastního algoritmu a posléze jeho implementace inspirovaná rozhodovacími stromy. Při analýze a testování byl rozhodovací strom upravován do finální podoby, která byla schopna vracet rozumné výsledky. Před samotnou implementací bylo třeba sehnat data, podle kterých algoritmus byl schopný rozpoznat chybu ve větě a následně predikovat a aplikovat její opravu. Textová data bylo třeba upravit do vhodného formátu, který by byl pro systém použitelný.

Finální podoba vytvořeného systému je schopna opravit základní pravopisné nebo gramatické chyby za předpokladu, že výsledná věta je obsažena v databázi. Pokud je míra chybovosti ve větě vysoká, systém vykazuje velké výstupní časy, které jsou pro praktické využití velice komplikované. Celý algoritmus je vypracován pro jazyk český. Jazyk český má volný slovosled, a to je mimo jiné jeden z mnoha důvodů, proč je pro tento jazyk tak náročné vytvořit automatickou korekci textu.

Pro dosažení lepších výsledků bychom potřebovali větší základnu vět po korekci a zároveň výkonnější konfiguraci hardwaru. Systém má velký potenciál při použití na velice výkonném zařízení s přístupem k neustálé aktualizující se datové základně s přibývajícím počtem vět, které jsou po korekci.

# Seznam zdrojů a použité literatury


[1] K. autorů, Pravidla českého pravopisu, Nakladatelství Universum, 2014.

[2] „MRC Cognition and Brain Sciences Unit," [Online]. Available: http://www.mrc-cbu.cam.ac.uk/people/matt.davis/cmabridge/.

[3] P. Brejčák, „Jak na prelkepy," [Online]. Available: https://www.zdrojak.cz/clanky/jak-na-prelkepy/.

[4] D. Čihák, Detekce překlepů v dotazech, Brno: MASARYKOVA UNIVERZITA, 2010.

[5] „N-gram," [Online]. Available: https://en.wikipedia.org/wiki/N-gram.

[6] „Levenshtein distance," [Online]. Available: https://en.wikipedia.org/wiki/Levenshtein_distance.

[7] K. Oliva, „http://docplayer.cz/," [Online]. Available: http://docplayer.cz/4387527-Jak-se-dela-gramaticky-korektor-cestiny-karel-oliva-ustav-pro-jazyk-cesky-akademie-ved-cr.html.

[8] V. Běhálková, Evaluace a srovnání nástrojů pro strojový překlad, Olomouc: Filozofická fakulta Univerzity Palackého, 2014.

[9] I. J. Kanis, STATISTICKÝ AUTOMATICKÝ PŘEKLAD, Plzeň: Západočeská univerzita v Plzni, 2009.

[10] „Statistický strojový překlad," [Online]. Available: https://blog.root.cz/babel/statisticky-strojovy-preklad/.

[11] „MS word umi ceskou gramatiku," [Online]. Available: http://weblog.jakpsatweb.cz/b/1120175412-ms-word-umi-ceskou-gramatiku.html.

[12] P. Ing. Petr Honzík, Strojové ucení, Brno: FEKT Vysokého ucení technického v Brne, 2006.

[13] „Microsoft documentation," [Online]. Available: https://docs.microsoft.com/en-us/sql/t-sql/statements/bulk-insert-transact-sql?view=sql-server-2017.

[14] [Online]. Available: https://www.novinky.cz/.

[15] „Slovník všech českých slov," [Online]. Available: http://forum.zive.cz/viewtopic.php?f=922&t=1204928.

[16] J. S. Y. Lee, Automatic Correction of Grammatical Errors, Massachusetts Institute of Technology, 2002.

[17] „Strojovy preklad," [Online]. Available: https://vesmir.cz/cz/casopis/archiv-casopisu/2012/cislo-9/strojovy-preklad.html.

# Adresářová struktura přiloženého disku

\app – obsahuje spustitelnou aplikaci
\source – obsahuje zdrojové kódy celého projektu